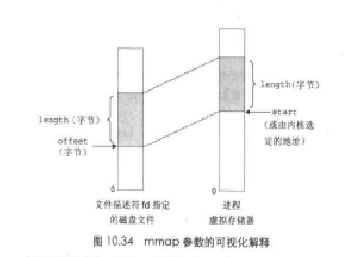
[mmap和shm共享内存的区别和联系](http://blog.chinaunix.net/uid-26335251-id-3493125.html) 2013-02-20 13:15:31

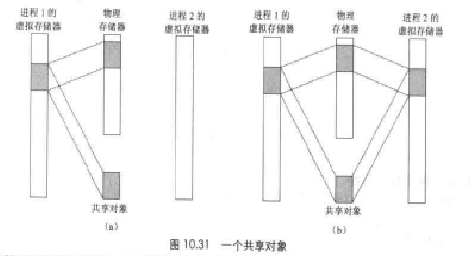
分类： LINUX

共享内存的创建

根据理论：

 1. 共享内存允许两个或多个进程共享一给定的存储区，因为数据不需要来回复制，所以是最快的一种进程间通信机制。共享内存可以通过mmap()映射普通文件（特殊情况下还可以采用匿名映射）机制实现，也可以通过系统V共享内存机制实现。应用接口和原理很简单，内部机制复杂。为了实现更安全通信，往往还与信号灯等同步机制共同使用。  
  
mmap的机制如：就是在磁盘上建立一个文件，每个进程存储器里面，单独开辟一个空间来进行映射。如果多进程的话，那么不会对实际的物理存储器（主存）消耗太大。  


shm的机制：每个进程的共享内存都直接映射到实际物理存储器里面。



结论：  
  
1、mmap保存到实际硬盘，实际存储并没有反映到主存上。优点：储存量可以很大（多于主存）（这里一个问题，需要高手解答,会不会太多拷贝到主存里面？？？）；缺点：进程间读取和写入速度要比主存的要慢。  
  
2、shm保存到物理存储器（主存），实际的储存量直接反映到主存上。优点，进程间访问速度（读写）比磁盘要快；缺点，储存量不能非常大（多于主存）  
  
使用上看：如果分配的存储量不大，那么使用shm；如果存储量大，那么使用shm。

参看百度：http://baike.baidu.com/view/1499209.htm

mmap就是一个文件操作

看这些百度的描述：

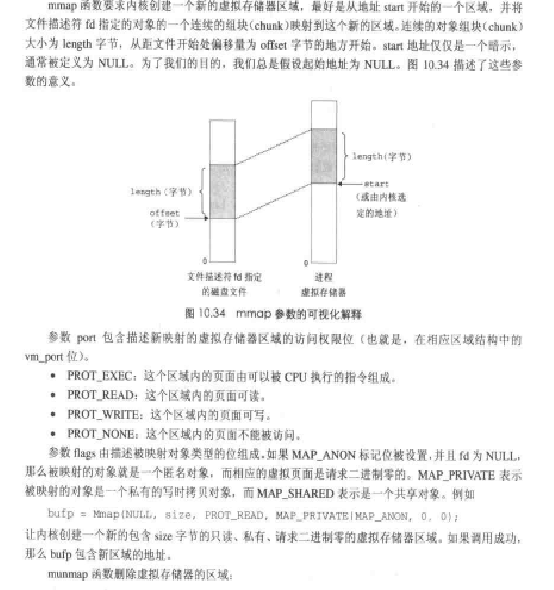
mmap()系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以向访问普通内存一样对文件进行访问，不必再调用read()，write（）等操作。 成功执行时，mmap()返回被映射区的指针，munmap()返回0。失败时，mmap()返回MAP\_FAILED[其值为(void \*)-1]，munmap返回-1。errno被设为以下的某个值 EACCES：访问出错EAGAIN：文件已被锁定，或者太多的内存已被锁定EBADF：fd不是有效的文件描述词EINVAL：一个或者多个参数无效 ENFILE：已达到系统对打开文件的限制ENODEV：指定文件所在的文件系统不支持内存映射ENOMEM：内存不足，或者进程已超出最大内存映射数量 EPERM：权能不足，操作不允许ETXTBSY：已写的方式打开文件，同时指定MAP\_DENYWRITE标志SIGSEGV：试着向只读区写入 SIGBUS：试着访问不属于进程的内存区参数fd为即将映射到进程空间的文件描述字，

一般由open()返回，同时，fd可以指定为-1，此时须指定 flags参数中的MAP\_ANON，表明进行的是匿名映射（不涉及具体的文件名，避免了文件的创建及打开，很显然只能用于具有亲缘关系的进程间通信）

相关文章参考：

mmap函数是unix/linux下的系统调用，来看《Unix Netword programming》卷二12.2节有详细介绍。

mmap系统调用并不是完全为了用于共享内存而设计的。它本身提供了不同于一般对普通文件的访问方式，进程可以像读写内存一样对普通文件的操作。而Posix或系统V的共享内存IPC则纯粹用于共享目的，当然mmap()实现共享内存也是其主要应用之一。  
          mmap系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以像访问普通内存一样对文件进行访问，不必再 调用read()，write（）等操作。mmap并不分配空间, 只是将文件映射到调用进程的地址空间里, 然后你就可以用memcpy等操作写文件, 而不用write()了.写完后用msync()同步一下, 你所写的内容就保存到文件里了. 不过这种方式没办法增加文件的长度, 因为要映射的长度在调用mmap()的时候就决定了.  
  
简单说就是把一个文件的内容在内存里面做一个映像，内存比磁盘快些。  
基本上它是把一个档案对应到你的virtual memory 中的一段，并传回一个指针。



重写总结：

1、mmap实际就是操作“文件”。

2、映射文件，除了主存的考虑外。shm的内存共享，效率应该比mmap效率要高（mmap通过io和文件操作，或“需要写完后用msync()同步一下”）；当然mmap映射操作文件，比直接操作文件要快些;由于多了一步msync应该可以说比shm要慢了吧？？？

3、另一方面，mmap的优点是，操作比shm简单（没有调用比shm函数复杂），我想这也是许多人喜欢用的原因，包括nginx。

缺点，还得通过实际程序测试，确定！！！

修正理解（这也真是的，这个网站没办法附加；只能重写了）：

今天又细心研究了一下，发现百度这么一段说明：

2、系统调用mmap()用于共享内存的两种方式：   
（1）使用普通文件提供的内存映射：适用于任何进程之间；此时，需要打开或创建一个文件，然后再调用mmap()；典型调用代码如下：   
fd=open(name, flag, mode);   
if(fd<0)   
...   
ptr=mmap(NULL, len , PROT\_READ|PROT\_WRITE, MAP\_SHARED , fd , 0); 通过mmap()实现共享内存的通信方式有许多特点和要注意的地方，我们将在范例中进行具体说明。   
（2）使用特殊文件提供匿名内存映射：适用于具有亲缘关系的进程之间；由于父子进程特殊的亲缘关系，在父进程中先调用mmap()，然后调用fork()。那么在调用fork()之后，子进程继承父进程匿名映射后的地址空间，同样也继承mmap()返回的地址，这样，父子进程就可以通过映射区域进行通信了。注意，这里不是一般的继承关系。一般来说，子进程单独维护从父进程继承下来的一些变量。而mmap()返回的地址，却由父子进程共同维护。

看了一下windows“内存映射文件”：<http://baike.baidu.com/view/394293.htm>

内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，只是内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而非系统的页文件，而且在对该文件进行操作之前必须首先对文件进行映射，就如同将整个文件从磁盘加载到内存。由此可以看出，使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行I/O操作，这意味着在对文件进行处理时将不必再为文件申请并分配缓存，所有的文件缓存操作均由系统直接管理，由于取消了将文件数据加载到内存、数据从内存到文件的回写以及释放内存块等步骤，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。另外，实际工程中的系统往往需要在多个进程之间共享数据，如果数据量小，处理方法是灵活多变的，如果共享数据容量巨大，那么就需要借助于内存映射文件来进行。实际上，内存映射文件正是解决本地多个进程间数据共享的最有效方法。

这里再总结一次：

1、mmap有两种方式，一种是映射内存，它把普通文件映射为实际物理内存页，访问它就和访问物理内存一样（这也就和shm的功能一样了）（同时不用刷新到文件）

2、mmap可以映射文件，不确定会不会像windows“内存映射文件”一样的功能，如果是，那么他就能映射好几G甚至好几百G的内存数据，对大数据处理将提供强大功能了？？？

3、shm只做内存映射，和mmap第一个功能一样！只不过不是普通文件而已，但都是物理内存。

希望大家出意见！！！